**Доклад о языке программирования Go**

**Введение**

Go, также известный как Golang, был разработан в Google в 2007 году и официально представлен в 2009 году. Этот язык был создан с целью улучшения продуктивности разработки программного обеспечения, а также для повышения производительности и упрощения процессов написания кода. Go сочетает в себе характеристики компилируемого языка, статической типизации и простоты синтаксиса, что делает его удобным для программистов.

**Основные характеристики языка Go:**

1. Простота и читабельность:

- Синтаксис Go очень простой и лаконичный, что упрощает написание и понимание кода.

2. Статическая типизация:

- Go является статически типизированным языком, что позволяет выявлять ошибки на этапе компиляции, а не во время выполнения программы.

3. Параллелизм:

- Встроенная поддержка параллелизма через горутины (goroutines) и каналы (channels) делает Go идеальным для создания высокопроизводительных приложений.

4. Компиляция:

- Go компилируется в машинный код, что обеспечивает высокую производительность и независимость от платформы.

5. Стандартная библиотека:

- Go имеет мощную стандартную библиотеку, которая включает множество пакетов для работы с сетями, веб-приложениями, обработки данных и многого другого.

6. Сборка мусора:

- Go включает встроенный механизм сборки мусора, который помогает управлять памятью и предотвращает утечки.

**Структура программы на Go**

Программа на Go состоит из пакетов, и каждая программа должна содержать пакет `main`, где начинается выполнение. В Go также используются функции, которые могут принимать аргументы и возвращать значения.

В основе языка Golang — база лучших функций из языков C и C++, [Python](https://blog.skillfactory.ru/glossary/python/), [Pascal](https://blog.skillfactory.ru/glossary/pascal/), Oberon и Modula. Сначала Go использовали внутри Google, но затем он стал применяться в компаниях по всему миру: HP, Adobe, Microsoft, BBC, Uber, Dropbox, Netflix, Яндекс, ВКонтакте, Avito, Ozon и других.

**Инструменты для разработчиков**

В Go есть инструменты, которые ускоряют разработку и помогают решать разные задачи:

* [**typecheck**](https://pkg.go.dev/github.com/grailbio/bigslice/typecheck)проверит соответствие типов в коде;
* [**gas**](https://pkg.go.dev/github.com/andrebq/gas)найдет уязвимости;
* [**go vet**](https://pkg.go.dev/github.com/golangci/govet)поможет обнаружить ошибки в коде;
* [**gofmt**](https://pkg.go.dev/github.com/visualfc/gotools/gofmt)правильно отформатирует код, проставит пробелы для выравнивания и табы для отступа;
* [**godoc**](https://pkg.go.dev/github.com/zquestz/s/providers/godoc) найдет комментарии и подготовит из них мануал к программе, и другие.

Также в Go от Google есть пакет [pprof](https://pkg.go.dev/net/http/pprof). Он позволяет узнать, какие фрагменты кода выполняются очень долго, где программа сильно нагружает процессор или занимает много памяти. Результат работы представлен в виде текстового отчета, профайла. Для его использования нужна утилита [graphviz](https://pkg.go.dev/github.com/goccy/go-graphviz).

**Наличие «сборщика мусора»**

Это алгоритм, который сканирует код, находит объекты, замедляющие его работу, и удаляет их. «Сборщик мусора» обеспечивает высокую скорость исполнения программ и эффективное использование ресурсов. В некоторых языках общего назначения «сборщиков мусора» нет и память приходится очищать вручную — как, например, в C++.

**Кроссплатформенность**

Язык от Google поддерживается на Windows, [Linux](https://blog.skillfactory.ru/glossary/linux/), macOS, Android. Также он работает с FreeBSD, OpenBSD и другими UNIX-системами. Код также обладает переносимостью: программы, написанные для одной из этих операционных систем, могут быть легко с перекомпиляцией перенесены на другую ОС.

**Нехватка конструкций для ООП**

Язык Go поддерживает не все возможности объектно-ориентированного программирования. Например, в нем нет классов и наследования. Это тоже намеренное решение для упрощения кода, чтобы добиться минимума избыточности и обеспечить высокую скорость исполнения программ.

**Пример кода**

В качестве примера, приведем простой HTTP-сервер на Go, который обслуживает HTTP-запросы и возвращает "Hello, World!".

```go

package main

import (

"fmt"

"net/http"

)

// Обработчик для корневого URL

func helloHandler(w http.ResponseWriter, r \*http.Request) {

fmt.Fprintf(w, "Hello, World!")

}

func main() {

// Устанавливаем обработчик для маршрута "/"

http.HandleFunc("/", helloHandler)

// Запускаем сервер на порту 8080

fmt.Println("Запуск сервера на :8080...")

if err := http.ListenAndServe(":8080", nil); err != nil {

fmt.Println("Ошибка запуска сервера:", err)

}

}

```

Объяснение кода

1. Импорт пакетов:

- В коде мы импортируем два пакета: `fmt` для форматированного ввода/вывода и `net/http` для работы с HTTP.

2. Определение обработчика:

- `helloHandler` — это функция, обрабатывающая HTTP-запросы и отправляющая текст "Hello, World!" в ответ.

3. Основная функция:

- В функции `main` мы устанавливаем маршрут для корневого URL (`/`) с помощью `http.HandleFunc` и запускаем HTTP-сервер на порту 8080 с помощью `http.ListenAndServe`.

4. Запуск сервера:

- Программа выводит сообщение о запуске сервера и обрабатывает входящие запросы.

**Заключение**

Go — это мощный язык программирования, который идеально подходит для создания серверных приложений, облачных решений и микросервисов. Его простота, эффективность и встроенная поддержка параллелизма делают его привлекательным выбором для разработчиков. С учетом постоянного роста популярности языка и активного сообщества, Go продолжает развиваться и внедряться в новые области разработки программного обеспечения.